developer.mozilla.org

Como estruturar um formulário HTML - Aprendendo desenvolvimento web | MDN

14-19 minutos

- Anterior (en-US)
- Menu: Forms (en-US)
- Próxima (en-US)

Com o básico fora do caminho, podemos olhar melhor para os elementos utilizados, a fim de entender as diferentes partes de um formulário.

A flexibilidade dos formulários HTML fazem com que sejam uma das estruturas mais complexas do HTML, você pode construir qualquer tipo básico de formulário usando elementos e atributos exclusivos de formulários. Assim, usar a estrutura correta ao criar um formulário HTML o ajudará a garantir que o formulário seja utilizável e acessível.

O elemento <form>

O elemento `<form>` é o elemento que formalmente define o formulário e os atributos que definem a
maneira como esse formulário se comporta. Sempre que você desejar criar um formulário HTML, você deve iniciá-lo usando este elemento, colocando todo o conteúdo dentro deste. Muitas

tecnologias assistivas ou plugins de navegadores são capazes de descobrir elemetos `<form>` e de implementar ganchos especiais para torná-los mais fáceis de usar.

Nós já vimos isto em um artigo anterior:

Nota: É estritamente proíbido aninhar um formulário dentro de outro formulário. Isto pode fazer com que os formulários se comportem de maneira imprevisível baseada no navegador que está sendo utilizado.

Note que, sempre é possível usar um widget de formulário fora de um elemento `<form>` mas se o fizer, o widget não terá nada a ver com o formulário. Estes widgets podem ser usados fora de um formulário, mas para tanto você deverá ter um plano especial para eles, pois este não farão nada por si próprios. Você terá de controlar o comportamento deles através de JavaScript.

Nota: O HTML 5 introduziu o atributo `form` no grupo de elementos de formulários em HTML. Ele deve deixá-lo atrelar explícitamente um elemento com um form mesmo se não estiver dentro de um `<form>`. Infelizmente, devido ao tempo de vida, essa implementação ainda pelos navegadores ainda não é boa o suficiente para se confiar nela.

Os elementos <fieldset> e <legend>

O elemento `<fieldset>` é uma maneira conveniente de criar grupos de widgets que compartilham o mesmo propósito. Um elemento `<fieldset>` pode ser rotulado com um elemento `<legend>`. O elemento `<legend>` descreve formalmente a finalidade do elemento `<fieldset>` ao qual está incluído.

Muitas tecnologias assistivas usarão o elemento `<legend>`

como se fosse parte da etiqueta de cada widget dentro do elemento <fieldset> correspondente. Por exemplo, alguns leitores de tela, como Jaws ou NVDA, pronunciarão o conteúdo da legenda antes de pronunciar o rótulo de cada widget.

Aqui está um pequeno exemplo:

```
<form>
  <fieldset>
    <legend>Fruit juice size</legend>
    <p>
      <input type="radio" name="size"
id="size_1" value="small" />
      <label for="size_1">Small</label>
    </p>
    <p>
      <input type="radio" name="size"
id="size_2" value="medium" />
      <label for="size_2">Medium</label>
    </p>
    <p>
      <input type="radio" name="size"
id="size_3" value="large" />
      <label for="size_3">Large</label>
    </p>
  </fieldset>
</form>
```

Com este exemplo, um leitor de tela pronunciará "Fruit juice size small" para o primeiro widget, "Fruit juice size medium" para o segundo, e "Fruit juice size large" para o terceiro.

O caso de uso neste exemplo é um dos mais importantes. Cada

vez que você tiver um conjunto de botões de opção, você deve ter certeza de que eles estão aninhados dentro de um elemento `<fieldset>`. Existem outros casos de uso e, em geral, o elemento `<fieldset>` também pode ser usado para conferir acessibilidade a um formulário. Devido à sua influência sobre a tecnologia assistiva, o elemento `<fieldset>` é um dos elementos-chave para a construção de formulários acessíveis. No entanto, é sua responsabilidade não abusá-lo. Se possível, cada vez que você criar um formulário, tente ouvir como o leitor de tela o interpreta. Se parecer estranho, é uma boa sugestão de que sua estrutura de formulário precisa ser melhorada.

O elemento <label>

Como vimos no artigo anterior, o elemento `<label>` é a maneira formal de definir um rótulo para um widget de formulário HTML. Esse é o elemento mais importante se você quiser criar formulários acessíveis - quando implementados corretamente, os leitores de tela falarão o rótulo de um elemento de formulário juntamente com as instruções relacionadas. Veja este exemplo, que vimos no artigo anterior:

```
<label for="name">Name:</label> <input
type="text" id="name" name="user_name">
```

Com o <label> associado corretamente ao <input> por meio de seus atributos 'for' e 'id' respectivamente (o 'label' do atributo faz referência ao atributo id do widget correspondente), um leitor de tela lerá algo como "Nome, editar texto".

Se o 'label' não estiver configurado corretamente, um leitor de tela só lerá algo como "Editar texto em branco", o que não é muito útil.

Observe que um widget pode ser aninhado dentro de seu

elemento <label>, assim:

```
<label for="name">
  Name: <input type="text" id="name"
name="user_name">
</label>
```

Mesmo nesses casos, entretanto, é considerada a melhor prática definir o atributo 'for' porque algumas tecnologias assistivas não entendem os relacionamentos implícitos entre labels e widgets.

Labels são clicáveis também !.

Outra vantagem de configurar 'labels' adequadamente é que você pode clicar no label para ativar o widget correspondente, em todos os navegadores. Isso é útil para exemplos como entradas de texto, onde você pode clicar no label, bem como na entrada para focalizá-lo, mas é especialmente útil para botões de opção e caixas de seleção - a área de acerto de tal controle pode ser muito pequena, então é útil para torná-lo o maior possível.

Por exemplo:

```
<form>
  <p>
    <label for="taste_1">I like cherry</label>
    <input type="checkbox" id="taste_1"
name="taste_cherry" value="1">
  </p>
  <p>
    <label for="taste_2">I like banana</label>
    <input type="checkbox" id="taste_2"
```

```
name="taste_banana" value="2">
  </p>
</form>
```

Múltiplos labels

Estritamente falando, você pode colocar vários rótulos em um único widget, mas isso não é uma boa ideia, pois algumas tecnologias de assistência podem ter problemas para lidar com eles. No caso de vários rótulos, você deve aninhar um widget e seus rótulos dentro de um único elemento `<label>`.

Vamos considerar este exemplo:

```
<p>Required fields are followed by <abbr
title="required">*</abbr>.</p>

<!-- So this: -->
<div>
  <label for="username">Name:</label>
  <input type="text" name="username">
  <label for="username"><abbr
title="required">*</abbr></label>
</div>

<!-- would be better done like this: -->
<div>
  <label for="username">
    <span>Name:</span>
    <input id="username" type="text"
name="username">
    <abbr title="required">*</abbr>
```

```
  </label>
</div>

<!-- But this is probably best: -->
<div>
  <label for="username">Name: <abbr
title="required">*</abbr></label>
  <input id="username" type="text"
name="username">
</div>
```

The paragraph at the top defines the rule for required elements. It must be at the beginning to make sure that assistive technologies such as screen readers will display or vocalize it to the user before they find a required element. That way, they will know what the asterisk means. A screen reader will speak the star as "*star*" or "*required*", depending on the screen reader's settings — in any case, what will be spoken is made clear in the first paragraph).

- In the first example, the label is not read out at all with the input — you just get "edit text blank", plus the actual labels are read out separately. The multiple `<label>`elements confuse the screenreader.
- In the second example, things are a bit clearer — the label read out along with the input is "name star name edit text", and the labels are still read out separately. Things are still a bit confusing, but it's a bit better this time because the input has a label associated with it.
- The third example is best — the actual label is read out all together, and the label read out with the input is "name star edit

text".

Note: You might get slightly different results, depending on your screenreader. This was tested in VoiceOver (and NVDA behaves similarly). We'd love to hear about your experiences too.

Note: You can find this example on GitHub as required-labels.html (see it live also). don't run the example with 2 or 3 of the versions uncommented — screenreaders will definitely get confused if you have multiple labels AND multiple inputs with the same ID!

Estruturas comuns HTML usadas com formulários

Além das estruturas específicas dos formulários HTML, é bom lembrar que os formulários são apenas HTML. Isso significa que você pode usar todo o poder do HTML para estruturar um formulário HTML.

Como você pode ver nos exemplos, é comum envolver um *label* e seu *widget* com um elemento `<div>`. Os elementos `<p>` também são comumente usados, assim como as listas HTML (a última é mais comum para estruturar vários *checkboxes* ou *radio buttons*).

Além do elemento `<fieldset>`, uma prática comum também é usar títulos HTML (por exemplo, `<h1>` (en-US), `<h2>` (en-US)) e seção (por exemplo, `<section>`) para estruturar um formulário complexo.

Acima de tudo, cabe a você encontrar um estilo com o qual você acha confortável codificar e que também resulte em formulários acessíveis e utilizáveis.

Temos cada seção desacoplada da funcionalidade contida nos elementos de `<section>` e um `<fieldset>` para conter os

radio buttons.

Active learning: building a form structure

Let's put these ideas into practice and build a slightly more involved form structure — a payment form. This form will contain a number of widget types that you may not yet understand — don't worry about this for now; you'll find out how they work in the next article (The native form widgets). For now, read the descriptions carefully as you follow the below instructions, and start to form an appreciation of which wrapper elements we are using to structure the form, and why.

1. To start with, make a local copy of our blank template file and the CSS for our payment form in a new directory on your computer.
2. First of all, apply the CSS to the HTML by adding the following line inside the HTML <head>:
   ```
   <link href="payment-form.css" rel="stylesheet">
   ```
3. Next, start your form off by adding the outer <form> element:
   ```
   <form>


   </form>
   ```
4. Inside the <form> tags, start by adding a heading and paragraph to inform users how required fields are marked:
   ```
   <h1>Payment form</h1>
   <p>Required fields are followed by
   <strong><abbr title="required">*</abbr>
   </strong>.</p>
   ```
5. Next we'll add a larger section of code into the form, below our previous entry. Here you'll see that we are wrapping the contact

information fields inside a distinct <section> element. Moreover, we have a set of two radio buttons, each of which we are putting inside its own list (<li>) element. Last, we have two standard text <input>s and their associated <label> elements, each contained inside a <p>, plus a password input for entering a password. Add this code to your form now:

```
<section>
    <h2>Contact information</h2>
    <fieldset>
      <legend>Title</legend>
      <ul>
          <li>
            <label for="title_1">
              <input type="radio" id="title_1"
name="title" value="M." >
              Mister
            </label>
          </li>
          <li>
            <label for="title_2">
              <input type="radio" id="title_2"
name="title" value="Ms.">
              Miss
            </label>
          </li>
      </ul>
    </fieldset>
    <p>
      <label for="name">
        <span>Name: </span>
```

```
        <strong><abbr title="required">*</abbr>
</strong>
      </label>
      <input type="text" id="name"
name="username">
    </p>
    <p>
      <label for="mail">
        <span>E-mail: </span>
        <strong><abbr title="required">*</abbr>
</strong>
      </label>
      <input type="email" id="mail"
name="usermail">
    </p>
    <p>
      <label for="pwd">
        <span>Password: </span>
        <strong><abbr title="required">*</abbr>
</strong>
      </label>
      <input type="password" id="pwd"
name="password">
    </p>
</section>
```

6. Now we'll turn to the second <section> of our form — the payment information. Here we have three distinct widgets along with their labels, each contained inside a <p>. The first is a drop down menu (<select>) for selecting credit card type. the

second is an <input> element of type number, for entering a credit card number. The last one is an <input> element of type date, for entering the expiration date of the card (this one will come up with a date picker widget in supporting browsers, and fall back to a normal text input in non-supporting browsers). Again, enter the following below the previous section:

```
<section>
    <h2>Payment information</h2>
    <p>
      <label for="card">
        <span>Card type:</span>
      </label>
      <select id="card" name="usercard">
        <option value="visa">Visa</option>
        <option value="mc">Mastercard</option>
        <option value="amex">American
Express</option>
      </select>
    </p>
    <p>
      <label for="number">
        <span>Card number:</span>
        <strong><abbr title="required">*</abbr>
</strong>
      </label>
      <input type="number" id="number"
name="cardnumber">
    </p>
    <p>
      <label for="date">
```

```
        <span>Expiration date:</span>
        <strong><abbr title="required">*</abbr>
</strong>
        <em>formatted as yyyy/mm/dd</em>
      </label>
      <input type="date" id="date"
name="expiration">
    </p>
</section>
```

7. The last section we'll add is a lot simpler, containing only a `<button>` of type `submit`, for submitting the form data. Add this to the bottom of your form now:

   ```
   <p> <button type="submit">Validate the
   payment</button> </p>
   ```

   You can see the finished form in action below (also find it on GitHub — see our payment-form.html source and running live):

Summary

You now have all the knowledge you'll need to properly structure your HTML forms; the next article will dig into implementing all the different types of form widgets you'll want to use to collect information from your users.

See Also

- A List Apart: *Sensible Forms: A Form Usability Checklist*
- Anterior (en-US)
- Menu: Forms (en-US)
- Próxima (en-US)

In this module